Trabalho Prático

Analisador Léxico e Tabela de Símbolos

O trabalho prático a ser realizado na disciplina de Compiladores é a construção de um compilador completo para a linguagem *Mini*. O trabalho será realizado por etapas, conforme cronograma a seguir. Este documento especifica as características da linguagem *Mini* e descreve as definições para a realização da primeira etapa do trabalho.

1. Cronograma e Valor

O trabalho vale 40% do total (tanto do primeiro quanto do segundo bimestre). Ele deverá ser entregue por etapas conforme cronograma abaixo:

Etapa	***Data de Entrega***	***Valor***
TP1 - Analisador Léxico e Tabela de Símbolos	13/10	20% (1ºB)
TP2 - Analisador Sintático	27/10	20% (1ºB)
TP3 - Analisador Semântico	16/11	20% (2ºB)
TP4 – Geração de Código	26/11	20% (2ºB)

2. Regras

O trabalho poderá ser realizado individual ou em grupo de até 3 pessoas.
A implementação deverá ser realizada em C/C++ ou Java ou Python ou LUA. O uso de outras linguagens está condicionado à aprovação do professor.
Poderão ser utilizadas ferramentas de geração de analisador léxico e sintático (sugeridos: *jflex* para léxico – http://jflex.de/download.html - e *cup* para sintático - http://www.cs.princeton.edu/~appel/modern/java/CUP/).
Realize as modificações necessárias na gramática para a implementação do analisador sintático.
Erros podem ser considerados fatais. Entretanto, a mensagens de erros correspondentes devem ser apresentadas, indicando a linha de ocorrência do erro.
Trabalhos total ou parcialmente iguais receberão avaliação nula.
A tolerância para entrega com atraso é de até 1 semana. Neste caso, incidirá multa na nota obtida no trabalho, conforme descrito na tabela acima. Não serão recebidos trabalhos após esse prazo.

3. Gramática da Linguagem *Mini*

program ::= **program** identifier body
body ::= [**declare** decl-list] **begin** stmt-list **end**
decl-list ::= decl {";" decl}
decl ::= type ident-list
ident-list ::= identifier {"," identifier}
type ::= **integer**
stmt-list ::= stmt {";" stmt}

stmt ::= assign-stmt | if-stmt | while-stmt
| read-stmt | write-stmt
assign-stmt ::= identifier ":=" simple_expr
if-stmt ::= **if** condition **then** stmt-list **end**
| **if** condition **then** stmt-list **else** stmt-list **end**
condition ::= expression
while-stmt ::= **do** stmt-list stmt-suffix
stmt-suffix ::= **while** condition
read-stmt ::= **read** "(" identifier ")"
write-stmt ::= **write** "(" writable ")"
writable ::= simple-expr | literal
expression ::= simple-expr | simple-expr relop simple-expr
simple-expr ::= term | simple-expr addop term
| "(" simple-expr ")" ? simple-expr ":" simple-expr
term ::= factor-a | term mulop factor-a
fator-a ::= factor | **not** factor | "-" factor
factor ::= identifier | constant | "(" expression ")"
relop ::= "=" | ">" | ">=" | "<" | "<=" | "<>"
addop ::= "+" | "-" | **or**
mulop ::= "*" | "/" | **mod** | **and**
shiftop ::= "<<" | ">>" | "<<<" | ">>>"
constant ::= digit {digit}
literal ::= " “" {caractere} "”"
identifier ::= letter {letter | digit}
letter ::= "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J"
| "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T"
| "U" | "V" | "W" | "X" | "Y" | "Z" | "a" | "b" | "c" | "d"
| "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l" | "m" | "n"
| "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x"
| "y" | "z"
digit ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
caractere ::= *um dos 256 caracteres do conjunto ASCII, exceto* "

4. Outras características de *Mini*

- As palavras-chave de *Mini* são reservadas.
- Toda variável deve ser declarada antes do seu uso.
- A semântica dos demais comandos e expressões é a tradicional do Pascal.
- Entrada e saída de *Mini* estão limitadas ao teclado e à tela de seu computador.
- Comentários de uma linha em *Mini* começam com %

5. Analisador Léxico e Tabela de Símbolos

Deverão ser entregues via submissão de arquivo no grupo do Teams grafo léxico (diagrama de transição) do analisador léxico (em PDF).

- Código fonte da Tabela de Símbolos.
- Código fonte do Analisador Léxico. No caso de uso de ferramenta, deverá ser entregue também o código fonte de entrada da ferramenta.
- Relatório contendo:

- Forma de uso do compilador
- Descrição da abordagem utilizada na implementação
- Resultados dos testes especificados a seguir. Os resultados deverão apresentar a saída do Analisador Léxico (a sequência de tokens identificados) e os símbolos instalados na Tabela de Símbolos.

6. Testes

Teste 1:

```
program teste1
declare
    integer a, b, c;
    integer result;
begin
    read (a);
    read (c);
    b := 10;
    result := (a * c)/(b + 5 % 345 -3) ;
    write(result)
end
```

Teste 2:

```
program teste2
declarando
    integer a;
    integer b;
    integer c;
begin
    read (a);
    b := a * a;
    c := b + a/2 * (3 + 5);
    write(c)
end;
```

Teste 3:

```
program teste3
declare
    integer id, qtd, cont, soma;
begin
    qtd := 5;
    cont := 5;
    soma := 0;
    do
        write("Idade: ");
        read (id);
        soma := soma # id;
        cont := cont - 1;
    while(cont > 0);
    write("Media: ");
    write (soma / qtd);
end
```

Teste 4:

```
program teste4
declare
    integer i, j, k, _var1;
begin
    i := 4 * (5-3) * -50 / 10;
```

```
    j := i * 10;
    k := i* j / k;
    k := -4 + 3 $;
    write(i);
    write(j);
    write(k);
end
```

Teste 5:

```
program teste5
% programa com if
declare
    integer j, k;
    begin
    read(j);
    read(k);
    if (k <> 0) then
        result := j/k
    else
        result := 0
    end
end
```

Teste 6:

```
program teste6
declare
    integer a, b, c, maior;
begin
    read(a);
    read(b);
    read(c);
    % identifica o maior
    if ( a>b and a>c ) then
        maior := a
    else
        if (b>c) then
            maior := b
        else
            maior := c
        end
    end;
    write(maior);
end
```

Teste 7:

```
program teste6
declare
    integer a, b, c, maior;
begin
    read(a);
    read(b);
    read(c);
    % identifica o maior
    maior:= (a>b and a>c)?a:(b>c)?b:c;
    write(maior);
end
```

Teste 8:

Mostre mais dois testes que demonstrem o funcionamento de seu analisador léxico.

****